INTERNET-DRAFT
Intended Status: Proposed Standard
Expires: September 2019

T. Herbert
Quantonium

March 8, 2019

IPv4 Extension Headers and UDP Encapsulated Extension Headers
draft-herbert-ipv4-udpencap-eh-01

Abstract

This specification defines extension headers for IPv4 and a method to encapsulate extension headers in UDP to facilitate transmission over the Internet, as well as a definition of an IPv4 flow label. The goal is to provide a uniform and feasible method of extensibility that is shared between IPv4 and IPv6.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright and License Notice

Table of Contents

1 Introduction

This specification defines extension headers for IPv4 and a method to encapsulate extension headers in UDP to facilitate transmission over the Internet. An IPv4 flow label is also defined to help intermediate nodes classify flows for packets with unknown protocols.

1.1 IPv4 extension headers

IPv4 options were defined in [RFC0791] as the means of extending the IP protocol. IPv4 options have not been successful. Early router implementations, and even those today, either don't process IPv4 options or relegate them to a slow path effectively making them unusable for serious applications. IPv4 options are limited to forty bytes length and, unlike TCP options, no IP options have been defined that are critical to communications. The upshot is that IPv4 options have long not been considered an option for deployment [IPNOPT].

IPv6 took a different approach. Extensibility of IPv6 is provided by extension headers. Optional internet-layer information is encoded in separate headers that may be placed between the IPv6 header and the upper-layer header in a packet [RFC8200]. IPv6 extension headers have had mixed success in deployment in that some intermediate devices have trouble processing them [RFC7872], however there are several active proposals in IETF that would make use of them (e.g. [FAST], [MTUOPT], [IOAM], [SRV6EH]).

This specification proposes that extension headers, those defined for IPv6, should be usable with IPv4 as a common method of extensibility. Using extension headers with IPv4 is logically straightforward. The IPv4 Protocol field is effectively re-designated to be a Next Header field with the same meaning and semantics as the IPv6 Next Header field. In this manner, an IPv4 packet can contain any defined IPv6 extension headers that are recast as IPv4 extension headers. These include Hop-by-Hop Options, Routing Header, Fragment, Destination Options, Authentication, and Encapsulating Security Payload. In cases

where an extension header contains IPv6 specific information, the extension header can be adapted for use with IPv4. For instance, a Routing Header carrying IPv6 addresses to visit could be adapted to carry IPv4 addresses.

1.2 Encapsulating extension headers in UDP

Deep Packet Inspection (DPI) is a common technique of middleboxes that has ossified Internet protocols in several ways. Attempts to use extension headers with IPv4 would likely be problematic for intermediate devices doing DPI. To address this, extension headers can be encapsulated in UDP using Generic UDP Encapsulation. The idea

is to insert a shim GUE/UDP header between an IPv4 (or IPv6) header and the extension headers. To nodes that don't understand extension headers, encapsulated extension headers are transparent and packets appear to be simple UDP/IP packets. To nodes that understand extension headers and the encapsulation, the GUE/UDP header is treated as an extension header itself that appears before any other extension headers.

Hop-by-Hop options are intended to be parsed, processed, and possibly modified by intermediate nodes in a path. When Hop-by-Hop options are encapsulated in UDP, consideration needs to be given on how to ensure robustness. Per [RFC7605], UDP port numbers only have meaning at the transport endpoints, so if an intermediate node attempts to interpret a UDP payload based solely on port number it may be incorrect. If a node were to modify a UDP payload whose type it has misinterpreted, then systematic silent data corruption ensues. To mitigate this issue, a magic number can be set in the UDP data that indicates the payload type. A magic number identifies the payload as being GUE with high probability to minimize the risk of misintepretation.

Note that the solution to encapsulate extension headers can be used for both IPv4 and IPv6. Encapsulation serves as workaround for paths that have problems processing IPv6 extension headers.

1.3 The IPv4 flow label

IPv6 introduced the concept of a flow label that has proven quite convenient to perform flow classification, such as that needed by Equal-Cost Multipath (ECMP). The base IPv4 header does not have

reserved bits that could be allocated as a flow label, however the sixteen bit Identification field can be used as a flow label in atomic datagrams [RFC6864].

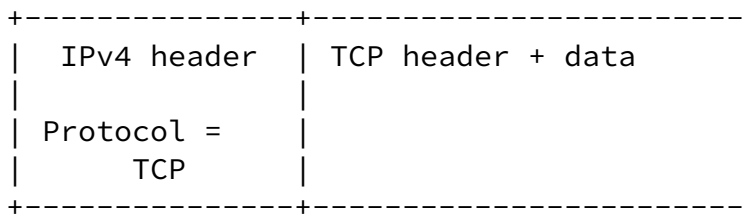
The IPv4 flow label will be most useful in scenarios for which the existing mechanisms used to classify IPv4 packets, such as parsing transport layer headers to extract port information, aren't available. Defining an IPv4 flow label would also be another instance of backporting a beneficial feature from IPv6 and further unifying the two protocols.

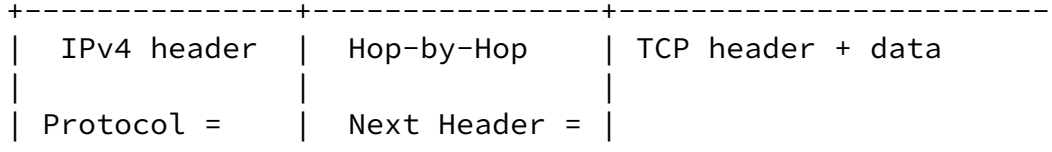
2 IPv4 extension headers

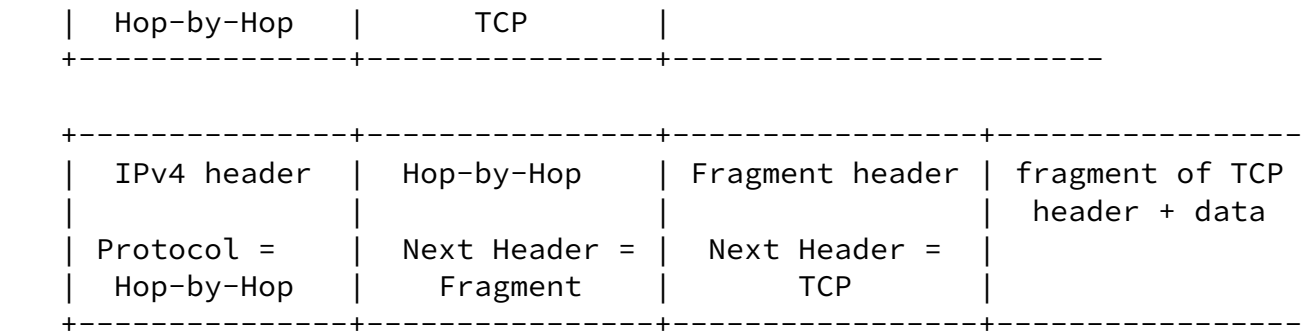
IPv4 extension headers are optional internet-layer information encoded in separate headers that may be placed between the IPv4 header and the upper-layer header in a packet. IPv4 extension headers are based on IPv6 extension headers and share the same basic properties and semantics [RFC8200].

Extension headers are numbered from IANA IP Protocol Numbers [IANA-PN], the same values are used for IPv4 and IPv6. When processing a sequence of Next Header values in a packet, the first one that is not an extension header [IANA-EH] indicates that the next item in the packet is the corresponding upper-layer header. A special "No Next Header" value is used if there is no upper-layer header.

As illustrated in these examples, an IPv4 packet MAY carry zero, one, or more extension headers, each identified by the Next Header field of the preceding header or the Protocol field of the IPv4 header:







2.1 Requirements

IPv4 extension headers normatively assumes the requirements of IPv6 extension headers as defined in [RFC8200] section 4, with the following modifications:

* References to the IPv6 header are replaced by references to the IPv4 header.

* ICMP errors sent in the course of processing extension headers use ICMPv4.

* The IPv4 header Protocol field assumes the same role and semantics with respect to extension headers as the IPv6 Next Header field.

* The Hop-by-Hop Options header is used to carry optional information that MAY be examined and processed by any node along a packet's delivery path.

* If a legacy IPv4 destination node, one that does not support IPv4 extension headers, receives a packet with extension headers then the packet will be processed as having an unknown protocol. It is expected that the packet will be discarded and an ICMP error is generated.

* Extension headers or options that carry IPv6 specific data or are otherwise specific to IPv6 MUST not be used with IPv4 (Segment Routing [SRV6EH] for example). IPv4 variants of these might be defined if achieving the same functionality in IPv4 is desirable.

* References to the IPv6 Payload Length, for instance in reassembly procedures, are interpreted as being the computed IPv4 payload length (i.e. IPv4 Total Length minus the length of the IPv4 header).

The following are modifications to fragmentation and reassembly requirements:

* References to setting the Payload Length field in the IPv6 header are interpreted to be setting the Total Length in the IPv4 header taking into account the IPv4 header length.

* When creating or modifying IPv4 headers in packets, the IPv4 header checksum MUST be set correctly.

* Different fragment packets MAY contain different IP options. The IP header and any options in the reassembled packet are taken from the first fragment packet (the one with offset of zero).

* If the length and offset of a fragment are such that the Total Length of the packet reassembled from that fragment would exceed 65,535 octets, then that fragment must be discarded and an ICMP Parameter Problem, Code 0, message should be sent to the source of the fragment, pointing to the Fragment Offset field of the fragment packet.

2.2 Interaction with standard IPv4 mechanisms

IPv4 extension headers may be used concurrently with IPv4 mechanisms such as IPv4 options and IPv4 fragmentation. This section discusses the interactions.

2.2.1 IPv4 options and IPv4 extension headers

An IPv4 packet MAY contain both IPv4 options and extension headers. IPv4 options are completely independent of IPv4 extension headers. IPv4 options MUST be processed before processing any extension headers per normal requirements of processing the IP header before the IP payload.

2.2.2 IPv4 fragmentation and IPv4 extension headers

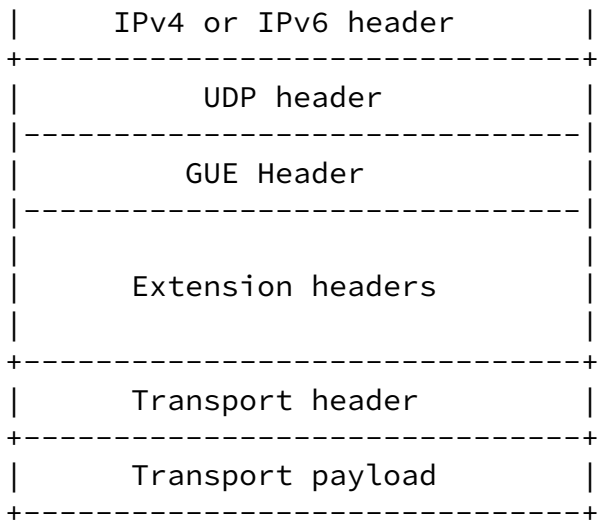
An IPv4 packet may be fragmented both by using a Fragment extension header as well as by standard IPv4 fragmentation. The Fragment header can only be set at the source, however intermediate devices can fragment packets using standard IPv4 fragmentation. Standard IPv4 fragmentation at a source node MUST be done only after any extension headers are set in a packet or the packet was fragmented using the Fragment header. Specifically, fragmentation using the extension header MUST NOT be done on packet fragments created by standard IPv4 fragmentation. However, a packet fragment that contains a Fragment header MAY itself be fragmented by standard IPv4 fragmentation. There is no correlation between normal IPv4 fragmentation and the IPv4 Fragment header, the identifier space for each are unrelated and reassembly procedures are independent.

At a destination, if a received packet was fragmented by standard IPv4 fragmentation, it MUST be reassembled before processing any IPv4 extension headers. This requirement ensures that standard IPv4 reassembly is done before reassembly for the Fragment header.

If an IPv4 packet containing Hop-by-Hop options is fragmented using standard IPv4 fragmentation, the Hop-by-Hop Options are not set in each of the packet fragments. An intermediate node MAY process the Hop-by-Hop options in the first fragment if the complete Hop-by-Hop extension header is contained within the fragment. If the Fragment header is used with IPv4 the DF bit (Don't Fragment) bit SHOULD be set in the IPv4 header and Path MTU discovery mechanisms SHOULD be used.

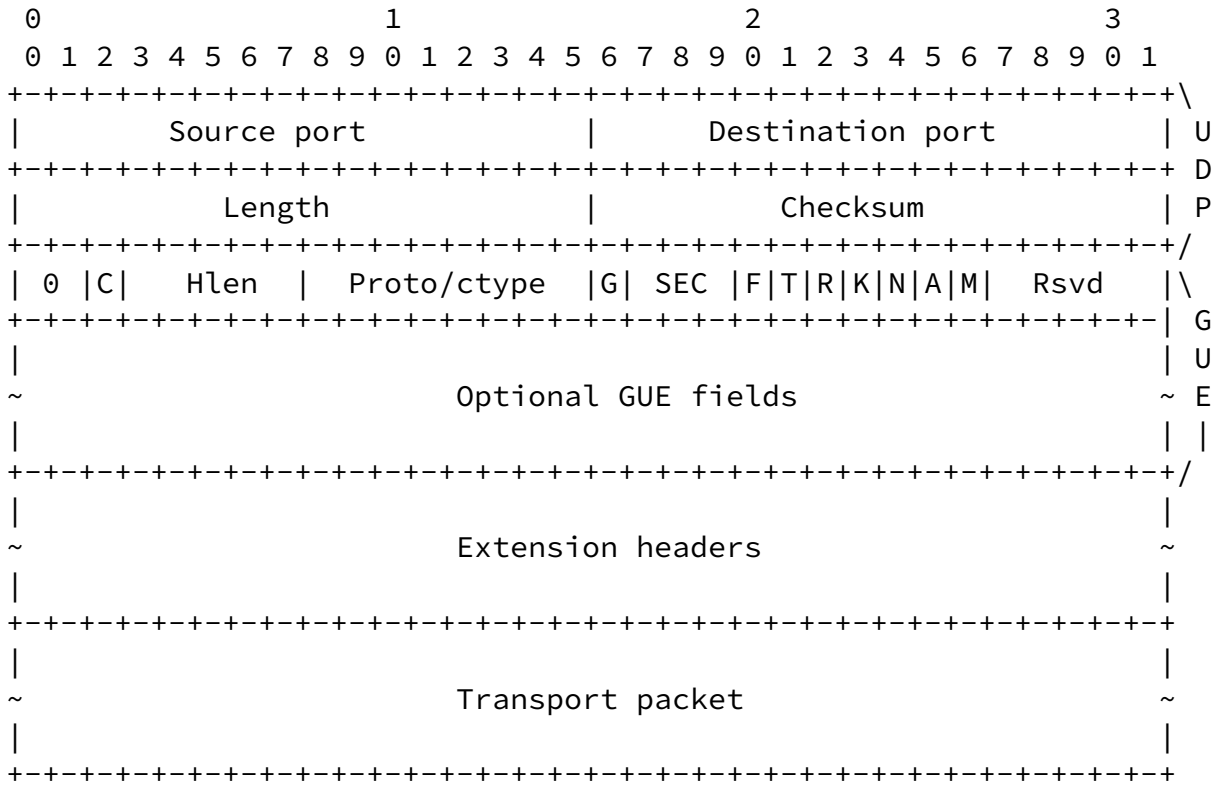
3 Encapsulating extension headers in UDP

This section defines a means to carry extension headers in Generic UDP Encapsulation (GUE). The diagram below illustrates the protocol stack when extension headers are encapsulated in GUE.



3.1 Encapsulation format

Extension headers and the trailing transport layer packet can be encapsulated in Variant 0 of GUE. The protocol format is:



The pertinent fields in the base GUE header are:

- Variant - set 0 for variant 0.

- C bit - Control bit. Set to zero indicating a data message.

- Hlen - Header length of GUE header in four byte words not including the first four bytes.

- Proto/ctype - The type of the encapsulated protocol. This is an IP protocol and may be an extension header. If the payload is something other than an IP protocol or the payload is encrypted or transformed, then this field is set to 59 (No Next Header)-- in this case the type of the payload is determined through other means.

- M: Magic number bit. If this bit is set then the GUE magic number option is present. The GUE magic number option is described below.

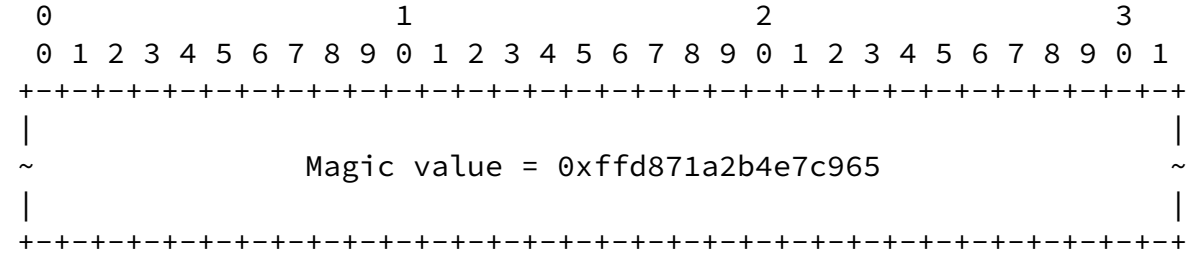
Any of the GUE options defined in [GUEEXT] MAY be set in the packet. To facilitate maintaining the correct transport layer checksum across NAT translation, the NAT address checksum option SHOULD be used ([GUEEXT]). The GUE magic number option, defined below, is used to help intermediate nodes correctly identify GUE packets.

If a transport layer protocol is encapsulated in GUE then the IP header for the transport header is taken to be the IP header of the GUE/UDP packet. In particular, an encapsulated transport header may have a checksum that includes the IP addresses in a pseudo header for checksum calculation (TCP or UDP).

3.2 GUE magic numbers

GUE magic numbers are used to identify a UDP payload as being a GUE payload with a high degree of probability.

The format of the GUE magic number option is:



The fields of the option are:

- Magic value. A 64 bit value that MUST be set to 0xffd871a2b4e7c965.

The GUE magic number option is present when the M bit is set in the GUE header flags.

3.3 Operation

This section describes the operation of encapsulating extension headers in GUE.

3.3.1 Sender processing

To encapsulate extension headers, a sender inserts a UDP and GUE header between an IP header and the first extension header.

If a sender encapsulates extension headers in GUE then it MUST NOT also set extension headers in the IPv4 or IPv6 header. When extension headers are encapsulated in GUE, the Next Header field of the IPv6 header or the Protocol field of the IPv4 header MUST be set to 17 to indicate UDP.

If the encapsulated transport protocol contains a checksum with a pseudo header and the packet may traverse a NAT, then the NAT Address Checksum option SHOULD be set to allow the receiver to properly adjust the received transport layer checksum. Other GUE options MAY be set per the discretion of the sender.

If the packet being encapsulated contains a Hop-by-Hop extension header then the Magic Number option MUST be used to allow intermediate nodes to process and potentially modify data in the extension header. Note that in this case the proto/ctype field in the GUE header MUST be zero indicating Hop-by-Hop options extension header.

The following guidelines apply to the source setting the magic number option:

- If the GUE checksum option is used then its payload coverage MUST be zero.
- If the GUE alternate checksum option is used then its payload coverage MUST be zero.
- If the HMAC security option is used then its Payload length MUST

be zero.

- The magic number option MUST NOT be set when the GUE fragmentation or payload transform option is used.

- The remote checksum option MAY be used concurrently with the magic number option under the assumption that intermediate nodes will not modify encapsulated transport checksum fields or attempt to verify an encapsulated transport layer checksum (in

the latter case they could do that it they were to take the remote checksum offload option into account).

3.3.2 Destination Processing

Encapsulated extension headers in GUE are processed by normal methods of processing GUE. As described in [GUE]:

> If a valid data message is received, the UDP header and GUE header are removed from the packet. The outer IP header remains intact and the next protocol in the IP header is set to the protocol from the proto field in the GUE header. The resulting packet is then resubmitted into the protocol stack to process that packet as though it was received with the protocol in the GUE header.

In the case that the GUE packet contains extension headers, the resultant packet after GUE processing is an IPv4 or IPv6 packet with extension headers. When the packet is resubmitted to the protocol stack, processing of the first extension header commences.

Note that if a routing header was encapsulated, the packet may be forwarded to another node. The packet MAY be re-encapsulated in GUE for transmission per the capabilities of the receiving node and network.

3.3.3 Intermediate device processing

Intermediate devices MAY process Hop-by-Hop options. In the case that GUE encapsulates Hop-by-Hop options, an intermediate node needs to parse, process, and possibly modify a UDP payload containing the GUE message with encapsulated Hop-by-Hop options. The magic number option is defined to allow intermediate nodes to identify GUE packets that

might contain Hop-by-Hop options to process.

Processing of packets with encapsulated Hop-by-Hop options has the following flow:

1) Match destination UDP port number to be GUE.

2) If the GUE variant is not zero or the C bit is set (control message) then discontinue payload processing.

3) If proto/ctype value is not zero (not Hop-by-Hop options) then discontinue payload processing.

5) If magic number option is not present in the GUE header then discontinue payload processing.

5) Compare the magic number value in the GUE header to the defined value. If they are not equal then discontinue payload processing

6) If the GUE checksum option is present (and payload coverage is zero) then the GUE checksum MAY be validated. If checksum validation fails, then discontinue payload processing

7) If the alternate checksum is present (and payload coverage is zero) then the alternate checksum MAY be validated. If alternate checksum validation fails, then discontinue payload processing

8) Process the encapsulated Hop-by-Hop options. If a Hop-by-Hop option is modified then the outer UDP checksum MUST be updated to reflect the change.

Note that an intermediate node MUST not modify any fields other then data in modifiable Hop-by-Hop options or the UDP checksum which needs to be updated when UDP payload is modified. In particular, intermediate nodes MUST NOT modify the GUE header nor an data aside from that in modifiable Hop-by-Hop options.

4 The IPv4 flow label

As stated in [RFC6864]:

> ">> Originating sources MAY set the IPv4 ID field of atomic datagrams to any value."

This specification allows the IPv4 ID to be used as a flow label in atomic datagrams. Atomic datagrams are IPv4 packets for which (DF==1)&&(MF==0)&&(frag_offset==0).

4.1 Sender requirements

An origin host MAY set the IPv4 Identification field as a flow label in atomic packets. The IPv4 flow label is set following the same procedures for setting the IPv6 flow label as described in [RFC6437], with the following modifications:

* The Identification field MUST NOT be used as a flow label in non-atomic fragments.

* Only stateless flow labels can be set.

* The value to set, e.g. from a hash computation over packet headers, is truncated to sixteen bits (the size of the

Identification field).

* If the IPv4 Identification field is not used as a flow label in atomic fragments, it SHOULD be set to zero.

* Intermediate nodes MUST NOT set the Identification field in atomic datagrams.

4.2 Receiver requirements

Receivers, including intermediate hosts, MAY process non-zero Identification fields in IPv4 header of atomic datagrams as being a flow label. The IPv4 flow label for instance can be used as input to ECMP as described in [RFC6438].

It is RECOMMENDED that a receiver only consumes the flow label if other typical means flow classification, such as parsing the

transport layer headers to extract port numbers for the flow, are not available. For instance, the IPv4 flow label could be used for flow based packet steering if a router encounters a packet with a protocol that is unknown to it.

5 Security Considerations

This specification enables use of IPv6 extension headers in IPv4. Related security mechanisms of IPv6 extension headers can be applied for use with IPv4 extension headers.

When extension headers are encapsulated in GUE, normal GUE security mechanisms can be used. If an intermediate node might modify GUE payload to process modifiable extension headers, then a GUE security algorithm cannot take input to authenticate the GUE payload. If authentication is necessary, then an Authentication header may be

used that treats modifiable data fields as zero-valued octets when computing or verifying the packet's authenticating value.

The IPv4 flow label has similar security properties as the IPv6 flow label. If the security intent of the sender is to prevent intermediate nodes in the network from classifying its traffic into flows then the IPv4 flow label SHOULD NOT be used.

6 IANA Considerations

IANA is requested to assign a value in the "GUE flag-fields" registry for the Magic Number option.

Flags bits	Field size	Description	Reference
Bit 10	8 bytes	Magic number	This document

7 References

7.1 Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC6864] Touch, J., "Updated Specification of the IPv4 ID Field", RFC 6864, DOI 10.17487/RFC6864, February 2013, <https://www.rfc-editor.org/info/rfc6864>.

[GUE] Herbert, T., Yong, L., Zia,)., "Generic UDP Encapsulation", draft-ietf-intarea-gue-07

7.2 Informative References

[RFC7872] Gont, F., Linkova, J., Chown, T., and W. Liu, "Observations on the Dropping of Packets with IPv6 Extension Headers in

the Real World", RFC 7872, DOI 10.17487/RFC7872, June 2016, <https://www.rfc-editor.org/info/rfc7872>.

[RFC7605] Touch, J., "Recommendations on Using Assigned Transport Port Numbers", BCP 165, RFC 7605, DOI 10.17487/RFC7605, August 2015, <https://www.rfc-editor.org/info/rfc7605>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <https://www.rfc-editor.org/info/rfc6437>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <https://www.rfc-editor.org/info/rfc6438>.

[IPNOPT] Rodrigo Fonseca, George Manning Porter, Randy H. Katz, Scott Shenker and Ion Stoica, "IP Options are not an option", <https://www2.eecs.berkeley.edu/Pubs/TechRpts/2005/EECS-2005-24.html>

[FAST] Herbert, T., "Firewall and Service Tickets", draft-herbert-fast-03

[MTUOPT] Hinden, R. and Fairhurst, G., "IPv6 Minimum Path MTU Hop-by-Hop Option", draft-hinden-6man-mtu-option-00

[IOAM] F. Brockners, S. Bhandari, V. Govindan, C. Pignataro, H. Gredler, J. Leddy, S. Youell, T. Mizrahi, D. Mozes, P. Lapukhov, R. Chang, "Encapsulations for In-situ OAM Data" draft-brockners-inband-oam-transport-05

[SRV6EH] C. Filsfils, Ed., S. Previdi, J. Leddy, S. Matsushima, D. Voyer, Ed., "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-16

[IANA-PN] IANA, "Protocol Numbers", <https://www.iana.org/assignments/protocol-numbers>.

[IANA-EH] IANA, "IPv6 Extension Header Types", <https://www.iana.org/assignments/ipv6-parameters>.

[GUEEXT] Herbert, T., Yong, L., and Templin, F., "Extensions for Generic UDP Encapsulation", draft-ietf-intarea-gue-extensions-07

Author's Address

Tom Herbert
Quantonium
Santa Clara, CA
USA

Email: tom@quantonium.net